Modification de la définition d'une table : La commande ALTER TABLE

Il est parfois nécessaire de modifier la structure d'une table, la commande ALTER TABLE sert à cela. Cette commande change la structure de la table mais pas son contenu.
Les types de modifications acceptées sont les suivants:

- ✓ Ajout d'une nouvelle colonne à la table avec ses contraintes
- ✓ Augmente ou diminuer la largeur d'une colonne existante
- ✓ Changer la catégorie d'une colonne, d'obligation à optionnelle ou vice versa (NOT NULL à NULL ou vice versa)
- ✓ Spécification d'une valeur par défaut pour une colonne existante
- ✓ Changer le type de données d'une une colonne existante
- ✓ Spécification d'autres contraintes pour une colonne existante
- ✓ Activer ou désactiver une contrainte
- ✓ Détruire une contrainte.

L'option ADD

Cette option permet d'ajouter une colonne ou une contrainte à une table existante.
Attention!!
Si la table contient des valeurs, alors la colonne ajoutée doit être mise à jour.
Si une contrainte est ajoutée à une colonne alors que celle-ci contient déjà des données qui ne correspondent pas à la contrainte, alors la modification de la structure de la table sera refusée.
Exemples :
Voici la commande CREATE initiale pour la table Employes

```
CREATE TABLE Employes(NumEmp number, nom varchar2(15), prenom
varchar2(20));
```

ALTER TABLE Employes ADD (Salaire NUMBER (8,2));

Permet d'ajouter la colonne Salaire à la table Employes

ALTER TABLE Employes ADD CONSTRAINT emppk PRIMARY KEY (NumEmp);

Permet d'ajouter une contrainte de clé primaire sur la colonne Numemp de la table Employes

L'option MODIFY:

Cette option permet de modifier le type de données, la valeur par défaut et la contrainte de NOT NULL sur une table déjà existante. Il est impossible de raccourci la taille d'une colonne (la longueur des données) si celle-ci contient des données.

ALTER TABLE Employes MODIFY (nom NOT NULL);

L'option ENABLE /DISABLE

Cette option sert à activer ou désactiver une contrainte.

ALTER TABLE Employes DISABLE Primary Key;

L'option DROP

Cette option sert à supprimer une contrainte sur une table déjà existante.

ALTER TABLE Employes DROP Primary Key;

Ou

ALTER TABLE Employes DROP CONSTRAINT emppk;

ALTER TABLE Employes DROP COLUMN nom;

ALTER TABLE Employes RENAME COLUMN Salaire TO SalaireEmp;

Supprimer une table : la commande <DROP TABLE>

La commande DROP permet de supprimer un objet de la b ase de données. (Table, indexe, synonyme..)

DROP TABLE personne;

Renommer une table : la commande <RENAME>

Permet de renommer une table ou un objet de la base de données

Syntaxe

```
RENAME <Ancien_nom> TO <Nouveau_nom>;
```

RENAME Employes TO EmployesInfo;

Requêtes avec jointures

Une jointure est une opération relationnelle qui sert à chercher des lignes ou des enregistrements à partir de deux ou plusieurs tables disposant d'un ensemble de valeur communes, en général les clés primaires.

Produit cartésien

Le produit cartésien est une requête de sélection qui met en jeux plusieurs tables. Pour deux tables, la sélection consiste à afficher la première ligne de la première table avec toutes les lignes de la deuxième table, puis la deuxième ligne de la première table avec toutes les lignes de la deuxième table et ainsi de suite. Ce type de sélection implique beaucoup de redondances.

Exemple

```
SELECT NOM, PRENOM, NOMPROG
FROM ETUDIANTS,PROGRAMME;
```

NOM	PRENOM	NOMPROG
PATOCHE	ALIAN	INFORMATIQUE
PATOCHE	ALIAN	ADMINISTRATION
PATOCHE	ALIAN	ELECTRONIQUE
FAFAR	CHANTALE	INFORMATIQUE
FAFAR	CHANTALE	ADMINISTRATION
FAFAR	CHANTALE	ELECTRONIQUE
MARTIN	RACHA	INFORMATIQUE
MARTIN	RACHA	ADMINISTRATION
MARTIN	RACHA	ELECTRONIQUE

…. (il y a une suite à la sortie de la requête)

Jointure simple :

Une jointure simple consiste est un produit cartésien avec un INNER JOIN faisant ainsi une restriction sur les lignes. La restriction est faîte sur l'égalité de la valeur de deux

attributs (cas de deux tables) qui sont généralement les clés primaires. (la valeur d'une clé primaire est égale à la valeur d'une clé étrangère)

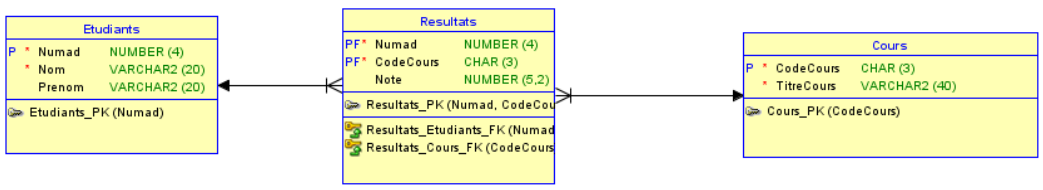
Exemple1

```
SELECT NOM, PRENOM, NOMPROG
FROM ETUDIANTS  INNER JOIN PROGRAMMES
ON Etudiants.CODEPRG =Programme.CODEPRG;
```

NOM	PRENOM	NOMPROG
PATOCHE	ALIAN	INFORMATIQUE
FAFAR	CHANTALE	INFORMATIQUE
MARTIN	RACHA	ADMINISTRATION
PLOUFFE	STEVEN	ELECTRONIQUE
ALLARD	MATHIEU	ADMINISTRATION
VIENS	NATHALIE	ADMINISTRATION
DANIS	SAMUEL	ELECTRONIQUE
FAVRE	NICOLAS	INFORMATIQUE
JACOB	YANICK	INFORMATIQUE

L'exemple précédent ramène des enregistrements uniquement s'il y a égalité entre la valeur de CODEPRG dans Etudiants est égale à la valeur de CODEPRG dans Programmes. Ce qui veut dire que les étudiants qui ont un CODEPRG null ne seront pas ramené par la requête.

Exemple 2




```
SELECT nom,prenom,  description, Resultats
FROM  ((etudiant E  INNER JOIN  Resultats R  ON E.numad = R.numad)
INNER JOIN cours C ON C.code_cours = R.code_cours);
```

Dans cet exemple on fait une jointure d'abord entre les tables Etudiants et Resultats par le numad, puis une jointure avec la table Cours par le code_cours.

Attention ! les tables ont des alias

- Lorsqu'un attribut sélectionné est présent dans plus d'une table alors il faut le précéder du nom de la table à partir de laquelle on désire l'extraire.
- Vous pouvez donner un alias aux noms de tables afin de faciliter la référence aux tables. Cependant si un alias est donné alors, il faudra utiliser l'alias à la place du nom de la table.
- Les attributs qui apparaissent dans la jointure ne sont pas nécessairement dans le SELECT.
- Toutes les tables dont les attributs apparaissent dans la clause SELECT ou dans la clause WHERE doivent apparaître dans la clause FROM.

Exemple 3 : **cette instruction va renvoyer une erreur car deptno est dans les deux tables Syemp et Sysdept.**

```
select ename, job, sal, loc
from (syemp inner join sydept on syemp.deptno = sydept.deptno) where deptno = 10;
```

Il faut écrire where sydept.deptno = 10;

Jointure Externe

Jointure externe droite : Dans la jointure externe droite, des enregistrements de table de la première table seront ramenés même si ceux-ci n'ont pas d'occurrences dans la deuxième table.

Jointure externe gauche : Dans la jointure externe gauche des enregistrements de table de deuxième table seront ramenés même si ceux-ci n'ont pas d'occurrences dans la première table.

Dans le cas d'une jointure externe, il faut faire suivre la colonne pour laquelle il n'est pas obligatoire d'avoir des lignes correspondant à l'égalité par l'opérateur LEFT OUTER JOIN ou RIGHT OUTER JOIN

Exemple
Cette requête ramène tous les étudiants y compris ceux qui ne sont pas inscrits dans un programme

```
SELECT NOM, PRENOM, NOMPROGRAMME
FROM ETUDIANTS E LEFT OUTER JOIN PROGRAMMEs P ON E.CODEP=P.CODEP;
```

	NOM	PRENOM	NOMPROGRAMME
1	Bouvier	Marge	Informatique
2	Simpson	bart	Informatique
3	Farar	Chantal	Informatique
4	Poirier	Juteux	Informatique
5	Patoche	Alain	Informatique
6	Saturne	Alain	Administration
7	Simpson	Christian	Administration
8	Patoche	Alain	Administration
9	Lefou	Duvillage	Santé animales
10	aaa	bbb	(null)
11	Bien	Thiery	(null)

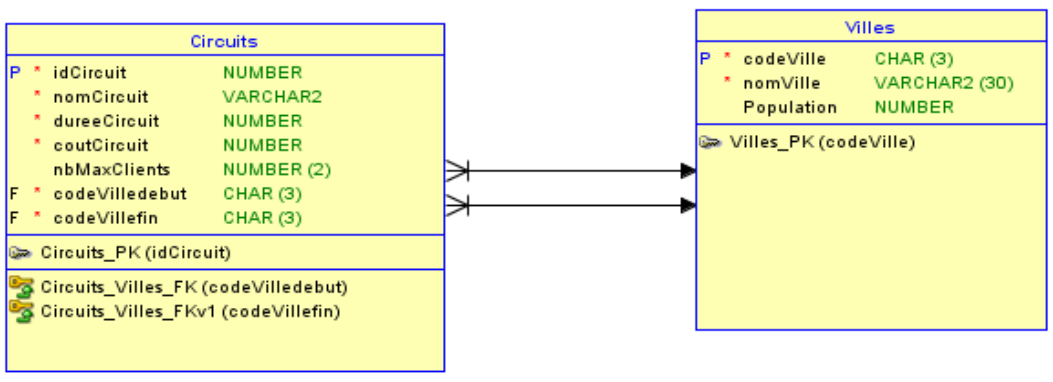
La requête suivante fait une jointure entre les deux tables Etudiants et Programmes. Elle ramène également les programmes qui n'ont pas d'étudiants.

```
SELECT NUMAD,NOM, PRENOM, NOMPROGRAMME
FROM ETUDIANTS E RIGHT OUTER JOIN PROGRAMMEs P ON E.CODEP=P.CODEP
ORDER BY NOM;
```

	NUMAD	NOM	PRENOM	NOMPROGRAMME
1	21	Bouvier	Marge	Informatique
2	12	Farar	Chantal	Informatique
3	14	Lefou	Duvillage	Santé animales
4	50	Patoche	Alain	Administration
5	10	Patoche	Alain	Informatique
6	11	Poirier	Juteux	Informatique
7	15	Saturne	Alain	Administration
8	13	Simpson	Christian	Administration
9	20	Simpson	bart	Informatique
10	(null)	(null)	(null)	AAAAAAAAAA
11	(null)	(null)	(null)	Génie industrielle
12	(null)	(null)	(null)	Hockey
13	(null)	(null)	(null)	Civilité

Importance des Alias :

Il arrive que la clé primaire d'une table migre plus qu'une fois dans une autre table pour être clé étrangère plus qu'une fois. L'exemple de la figure suivante montre que l'attribut codeVille est clé étrangère deux fois dans la table Circuits. Une fois pour indiquer le code ville de début du circuit, une autre fois pour indiquer le code ville de la fin du circuit.



Lors de la création de la table Circuits, nous devons mettre en évidence cette réalité de la clé étrangère deux fois sur la même clé primaire. Dans ce cas, il faudra donner des noms différents aux attributs de la clé étrangère. (On se rappelle qu'il faudra par contre qu'ils soient de même type de données et de même taille que l'attribut de la clé primaire)

```
create table Villes
(
codeVille char(3) constraint pkville primary key,
nomVille varchar(40) not null,
poulation number(8,0)
);


create table Circuits
(
idCircuit number(4,0),
nomCircuit varchar2(40),
codeVilledebut char(3),
codeVillefin char(3),
coutCircuit number(6,2),
dureeCircuit number(4,0),
constraint pkcircuit primary key (idCircuit),
constraint fkvileD Foreign key (codeVilleDebut) references villes(codeVille),
constraint fkvileA Foreign key (codeVilleFin) references villes(codeVille)
);
```

On veut écrire la requête qui ramène les circuits avec le nom des villes début du circuit et les noms des villes fin du circuit.
Comme il y a deux clé étrangères dans la table Circuits sur le même attribut de la table Villes, il faudra faire deux jointures sur la table Villes. C'est comme si nous avions deux tables Villes, une qui pour les débuts de circuits et l'autre pour la fin des circuits.

Dans la première jointure – FROM-- sur la ville nous avions donné l'alias De pour la table villes (pour départ) pour chercher le nom de ville de départ. Et pour la deuxième jointure sur la villes nous avion donné l'alias Ar pour la table villes (pour Arrivée).

Dans le SELECT, l'alias sur l'attribut nomville avec le mot AS n'Est pas obligtoire mais il sert à distinguer la ville de départ de la ville fin.

```
select nomCircuit, De.nomville as VilleDepart ,Ar.nomVille As villeArrivee,
coutCircuit,dureeCircuit
from (
(Villes De inner join Circuits on Circuits.CODEVILLEDEBUT = De.Codeville)
inner join villes Ar on Circuits.CODEVILLEFIN = Ar.codeville);
```